Individual Final Report

Xiaochi (George) Li

1. Introduction

This project is about using deep learning technique to de facial expression recognition.

2. My Individual Work

1. Data collection and literature review
2. Design and implement the structure of model training and the real time recognition.
Code includes:
    Code/Prototype/V2_main.py
    Code/Prototype/V2_helper.py
    Code/real_time_detection/RTD_V2.py
    Code/real_time_detection/RTD_helper.py
3. Design and implement Light VGG.
Code includes:
    Code/Prototype/Xiaochi_models.py
4. Design and implement transfer learning (Failed)
Code includes:
    Code/Prototype/transferlearning_models.py

3. My Individual Work in Detail

1. Data collection and literature review
The reason I choose this topic as our final project is that the facial expression recognition is a hot topic in Computer Vision and deep learning. You can begin with very basic MLP or CNN model to see some result, and you can also try the most advanced networks if you want.
I choose RAF-DB because each image has been tagged for 40 times which means the label should be quite good. Also, the image size is 100x100, which means we can use more convolution layers. We can even create a live demo of our work.

2. Design and implement the structure of model training and the real time recognition
The principle of design the structures is to make it easy to use and let the user focus on model design without worry other parts. I split the model definition part and data loading, performance plotting part out of the main program so that my teammates can easily define their model in the model definition part and get the evaluation plot in the end.
The principle of implementation is to make the code easy to read and modify and reduce time cost. One improvement I made in data loading part is to use pre-defined Numpy Array and plug in the

data instead of using append every time. This modification significantly reduced time complexity from O(n^2) to O(n).

And for the real time recognition part, it wasn't difficult, so I won't discuss it here.

3. Design and implement Light VGG
Please refer Final project group report section:
3.4 Light VGG One CNN structure inspired by VGGnet: Discussed the idea of Light VGG
"The original purpose of VGGnet is to be trained on the ImageNet and classify 1000 categories of objects from 224x224x3 input. And it's more complicated than our purpose: to classify 7 expression labels from 100x100x3 input. So, we decided to simplify the structure of original VGGnet to increase its speed and avoid overfitting while keep using multiple 3x3 convolution layer in sequence and increase the number of kernels as the network goes deeper"

4.4 Light VGG: Discussed structure of Light VGG.
Light VGG learns from VGG, but reduce the reduce the number of convolution blocks, the number of kernels in each convolution block and the size of fully connected layer after the convolution blocks

5.3 Light VGG: Discussed the performance result of Light VGG.
"Both of the network can be trained in a short time(~3 minutes), but they still have the problem of overfitting. And the accuracy stuck at 74%, after which the loss of the validation set rises again and indicates overfitting."

4. Design and implement transfer learning (Failed)
I tried to use transfer learning to improve model accuracy, however, I met an unexpected bug
"FailedPreconditionError: Attempting to use uninitialized value training/TFOptimizer/beta1_power"
I tried many ways to initialize all the variables but I can't solve this bug. I think the bug may related to some incompatible between Tensorflow and Keras.

4. Results

Both of the network can be trained in a short time(~3 minutes), but they still have the problem of overfitting. And the accuracy stuck at 74%, after which the loss of the validation set rises again and indicates overfitting. Thus, we need to apply early stop technique or check point the result of each epoch.

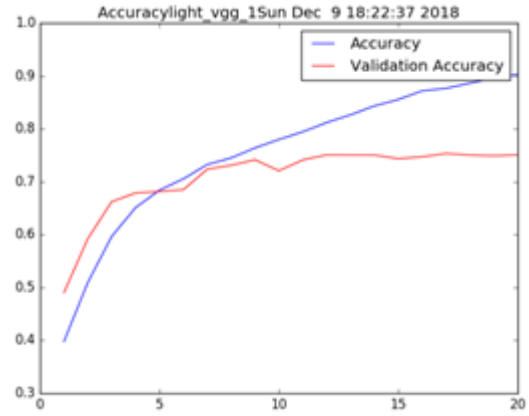
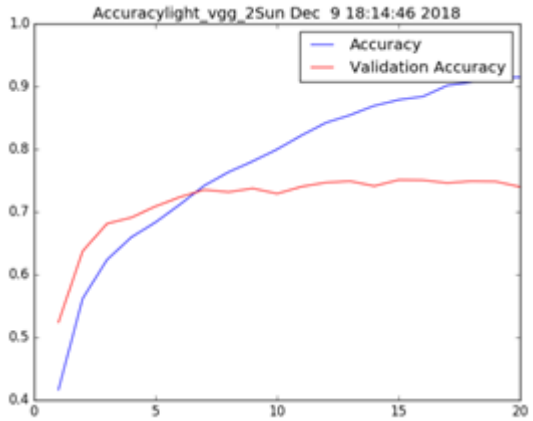
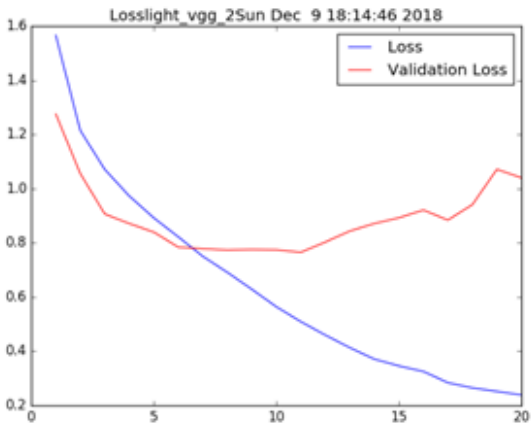
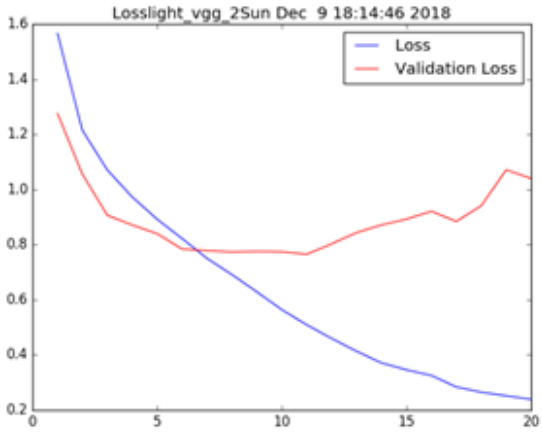
Light VGG V1	Light VGG V2
Accuracy	
	
Loss	
	

Table 8. Validation Accuracy and Validation Loss of Light VGG

5. Summary and conclusions

We have tried Multi-Layer Perceptron model and Convolution Neural Network in this project. The result suggests that MLP is not a good model for expression classification on RAF-DB, as the accuracy is only 67.8. Which has been consistent with the conclusion of many computer vision researchers.

One of the method to classify RAF-DB is to build a CNN from scratch and doing experiment on the structure and optimizers. And another is to learn from the common CNN structures and adopt their design method.

However, in this project, both method did not generate a perfect model. The accuracy is around 74%, and both models has unimprovable overfitting issue.
Despite the same accuracy and overfitting issue, light VGG V2 is our best model for RAF-DB single label subset because it's faster and smaller thus more suitable for real time facial expression detection.
After doing some literature review, the future step could be collecting more data, pretrain the model on other dataset (e.g. ImageNet) and use transfer learning.

We also made a prototype program of real time facial expression recognition to make the outcome of this project more practical instead of just comparing the accuracy and loss. We learnt that while the forward propagation in training can be very fast with CUDA on a powerful cloud VM, we still need to improve the model and reduce the time complexity to make it an industry level product.

6.Percentage of original code

Code/Prototype/V2_main.py :100%
Code/Prototype/V2_helper.py: 100%
Code/real_time_detection/RTD_V2.py:40/60 = 66%
Code/real_time_detection/RTD_helper.py:100%
Code/Prototype/Xiaochi_models.py: 100%
Code/Prototype/transferlearning_models.py:100%